Apple Assembly Line $1.80

Volume 6 -- Issue 5 February, 1986

Bag of Tricks 2

You've been asking when Bag of Tricks, that very popular and useful disk utility package, will be updated for ProDOS. Well, you can relax now: it's here.

The new ZAP program in "Bag of Tricks 2" adds the ability to access ProDOS blocks, directories, and files; the 80-column display can show most of a block at one view. The new version of FIXCAT can reconstruct a blown ProDOS directory, as far as is possible. You do still need to follow up with ZAP to correct things like file size and load address, which completely disappear when a directory is damaged.

This new, non-copy-protected edition of an old friend costs $49.95, or $45 + shipping from S-C. Owners of the older Bag of Tricks can get an upgrade directly from Quality Software for only $20 by returning your original disk.

Correction to Day of Week Programs

On page 20 of the December 1985 AAL, change lines 130 and 140 to the following:

```
     130 FOR I=0 TO 11 : READ MD(I) : NEXT
     140 FOR I=0 TO 6  : READ D$(I) : NEXT
```

On page 24, same issue, change line 120 to:

```
     120 FOR I=0 TO 6 : READ D$(I) : NEXT
```

That's what we get for typing a program into the Word Processor rather than printing a LISTing!

WildCAT for DOS 3.3.................................Erv Edge

WildCAT is a series of patches to DOS 3.3 which modify the CATALOG command. The new features include:

* A catalog by "wildcard" FILENAME facility.
* A catalog by FILETYPE facility.
* An alternate, short-form: either DIR or CAT.
* Catalog free space patch.
* Ctrl-Q catalog abort.
* TYPE a random or sequential text file.

Lee Reynold's FILEDUMP command has been re-packaged and re-presented as TYPE (see Call-A.P.P.L.E. 6/82 p47). More on this later. WildCAT, along with TYPE, is an attempt to teach new tricks to an old dog, as it were.

The normal DOS CATALOG command allows slot, drive, and volume parameters. I have added a filename parameter, but process it a little differently than the way file names are usually processed. To display the catalog entries for all files whose names contain a particular string, type any of the following:

```
CATALOG ^string [,Dn] [,Sn] [,Vn]
DIR ^string [,Dn] [,Sn] [,Vn]
CAT ^string [,Dn] [,Sn] [,Vn]
```

where "^string" begins with the "^" or caret symbol (shifted N on the][+ or shifted 6 on the //e or //c); the string should contain no blanks, although it may "end" with them; the string would normally end with a carriage return or with a comma if a drive or slot number is specified. Only those files that contain the "string" somewhere in the filename will be listed. (Of course you already know that the D, S, and V parameters are shown in brackets above because they are optional; you do not type the brackets.)

For example, "CATALOG ^TEST" would list each file with 'TEST' as part of the filename; while "DIR ^PAY." would list those with 'PAY.' somewhere in the filename; and "CAT^.OBJ,D2" would list filenames on drive 2 that contain the partial string '.OBJ'. "CAT" and "DIR" are simply synonyms for "CATALOG".

I have also arranged things so you can list the catalog entries of a specified file-type. You simply type the file type code in the CATALOG command:

```
CATALOG t [,Dn] [,Sn] [,Vn]
DIR t [,Dn] [,Sn] [,Vn]
CAT t [,Dn] [,Sn] [,Vn]
```

where "t" is any of the unadorned, single-letter filetypes: A B I R S T. Only that type of file (if present) will be listed.

For example, "CATALOG T" would list all the text files; "DIR A,D2" would list all of the Applesoft files on drive 2; "CAT B,S5,D1" would list all the binary files on slot 5, drive 1. Yes, "DIRT" works just fine.

S-C Macro Assembler Version 2.0	DOS $100, ProDOS $100, both for $120	
ProDOS Upgrade Kit for Version 2.0 DOS owners	$30	
Version 2.0 Upgrade Kit for 1.0/1.1/1.2 owners	$20	
Source Code of S-C Macro 2.0 (DOS only)	additional $100	
Full Screen Editor for S-C Macro (with complete source code)	$49	
S-C Cross Reference Utility	without source code $20, with source $50	
RAK-Ware DISASM	$30	
Source Code for DISASM	additional $30	
S-C Word Processor (with complete source code)	$50	
DP18 Source and Object	$50	
Double Precision Floating Point for Applesoft (with source code)	$50	
"Bag of Tricks", Worth & Lechner, with diskette	($39.95) $36	*
"Bag of Tricks 2"	($49.95) $45	*
MacASM -- Macro Assembler for MacIntosh (Mainstay)	($150.00) $100	*
S-C Documentor (complete commented source code of Applesoft ROMs)	$50	
Source Code of //e CX & F8 ROMs on disk	$15	
Cross Assemblers for owners of S-C Macro Assembler	$32.50 to $50 each	

(Available: 6800/1/2, 6301, 6805, 6809, 68000, Z-80, Z-8, 8048,
8051, 8085, 1802/4/5, PDP-11, GI1650/70, others)

AAL Quarterly Disks........................each $15, or any four for $45
Each disk contains the source code from three issues of AAL, saving you lots of typing and testing.
The quarters are Jan-Mar, Apr-Jun, Jul-Sep, and Oct-Dec.
(All source code is formatted for S-C Macro Assembler. Other assemblers require some effort to convert file type and edit directives.)

Diskettes (with hub rings)	package of 20 for $20	*
Vinyl disk pages, 6"x8.5", hold two disks each	10 for $6	*
Diskette Mailing Protectors (hold 1 or 2 disks) (Cardboard folders designed to fit 6"X9" Envelopes.)	40 cents each or $25 per 100	*
Envelopes for Diskette Mailers	6 cents each	

65802 Microprocessor (Western Design Center)	($95)	$50	*
quikLoader EPROM System (SCRG)	($179)	$170	*
PROmGRAMER (SCRG)	($149.50)	$140	*
Switch-a-Slot (SCRG)	($179.50)	$170	*
Extend-a-Slot (SCRG)	($35)	$32	*

"Programming the 65816", Eyes	($22.95)	$21	*
"Apple //e Reference Manual", Apple Computer	($24.95)	$23	*
"Apple //c Reference Manual", Apple Computer	($24.95)	$23	*
"ProDOS Technical Reference Manual", Apple Computer	($29.95)	$27	*
"Now That You Know Apple Assembly Language...", Gilder	($19.95)	$18	*
"Apple ProDOS: Advanced Features for Programmers", Little	($17.95)	$17	*
"Inside the Apple //c", Little	($19.95)	$18	*
"Inside the Apple //e", Little	($19.95)	$18	*
"Apple II+/IIe Troubleshooting & Repair Guide", Brenner	($19.95)	$18	*
"Apple][Circuit Description", Gayler	($22.95)	$21	*
"Understanding the Apple II", Sather	($22.95)	$21	*
"Understanding the Apple //e", Sather	($24.95)	$23	*
"Enhancing Your Apple II, vol. 1", Lancaster	($15.95)	$15	*
"Enhancing Your Apple II, vol. 2", Lancaster	($17.95)	$17	*
"Assembly Cookbook for the Apple II/IIe", Lancaster	($21.95)	$20	*
"Beneath Apple DOS", Worth & Lechner	($19.95)	$18	*
"Beneath Apple ProDOS", Worth & Lechner	($19.95)	$18	*
"Real Time Programming -- Neglected Topics", Foster	($9.95)	$9	*
"Microcomputer Graphics", Myers	($12.95)	$12	*
"Assem. Language for Applesoft Programmers", Finley & Myers	($16.95)	$16	*
"Assembly Lines -- the Book", Wagner	($19.95)	$18	*
"AppleVisions", Bishop & Grossberger	($39.95)	$36	*

* On these items add $2.00 for the first item and
$.75 for each additional item for US shipping.
Foreign customers inquire for postage needed.
Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
*** (214) 324-2050 ***
*** Master Card, VISA, Discover and American Express ***

I added the TYPE command, which allows you to display the contents of text files. Both CATALOG and TYPE will optionally:

1. Print "hidden" control characters as inverse:
   POKE 234,0 to print as inverse (default)
   POKE 234,255 to function as-is

2. Lower case letters may be shifted to upper case:
   POKE -18700,255 no shift (default)
   POKE -18700,223 to shift lower to upper case.

You can slow down TYPE's output via SPEED=xx or POKE 241,xx; or pause by pressing any key; then Ctrl-Q to abort. Also, TYPE pauses and waits for a keypress when it encounters a hex 00 imbedded in the file or at end of file; press Ctrl-Q to quit. You may TYPE random text files by holding down REPT-SPACE to get past the hex 00's at the end of each logical record.

The listing that follows is intended for information only: it is not BRUNable. My intention is that you prepare the EXEC shown below to actually install the patches. Any word processor that produces a straight, sequential text file may be used to prepare the EXEC. Of course you can also use the S-C Macro Assembler for this purpose. Then, type EXEC WILDCAT to apply the patches to DOS 3.3 in memory. After checking it out and running any other tests you like, put in a new diskette, enter a HELLO program, and type INIT HELLO to "permanently" install WildCAT in the DOS on tracks 0, 1, and 2.

When I wrote WildCAT, I had two main goals in mind: it should be a (mostly in-place) code replacement, and it should be compatible with the known means of using (abusing?) the existing CATALOG code at $AD98-AE69.

One major design consideration was a mechanism for entering the ^string/type parameter. This required merely changing the "keyword parameter table" so CATALOG could have a "filename".

Next, a distinction had to be made between a "wildcard" and a "filetype" parameter. It made sense to 'delimit' the wildcard string; then the single-character filetype would be just that: a single character, entered without a delimiter. But this "phony" name mechanism has it's own problems:

First, "What's in a Name?" (DOS Manual p. 16): a filename has to start with a letter...which automatically eliminates most special characters (eg, equal, pound, slash, colon, etc) as the delimiter. The command parsing routine doesn't really know what it's working on at the time. All it knows is: if a name may be present, it must be valid. The validity test is only that the first character be equal to or greater that $C0 or an @-sign. The @-sign could have been used, but it's a problem on some 80-column boards; the ^ or caret works nicely (and besides, it looks good).

Second, now that we have a name (however, phony) and since the CATALOG command lives in the File Manager (FM) portion of DOS, there will be a buffer allocated for it. Unfortunately, the

Command Interpreter (CI) DOCAT routine, which calls the FM, already "knows" that there will not really be a name, so it does not include housekeeping code to deallocate a buffer. So merrily allocating files without closing them...after the third time: NO BUFFERS AVAILABLE. And naively adding CLOSE (even if there were room for it), would have one very undesirable side effect if a "regular" catalog were requested: CATALOG-CLOSE without FNAME will close all open files. WildCAT instead plays a little shell game with DOS: The new DOCAT routine saves the first character of FNAME and substitutes a zero. Thereafter, neither the File Manager nor the rest of DOS ever knows that a name has been entered, so FM never actually allocates a buffer.

Third, what really should happen if a phony name is not entered? A regular catalog, of course, but how would this be indicated to WildCAT? Well, the shell game has a sting. Early on when the CI PARSE routine discovers that a filename is a valid parameter, it first clears FNAME to all blanks, expecting to fill it in with whatever comes in next. If a comma or carriage return comes in next, then FNAME still contains the blank; and that's what WildCAT saves off (under the shell) before it substitutes the zero.

Thus, the "sting" is that the CI "tricks" itself into telling WildCAT what to do in the absence of a ^string/type specifier: WildCAT takes a blank to indicate "do a regular" catalog; just as positively as a "^" indicates "do a wildcard" catalog, and a single character indicates "do a filetype" catalog.

The blank indicator also helps satisfy the second goal above and solves the problem of compatibilty with the "known means" of using/abusing the existing CATALOG code. WildCAT simply has to put a blank under the shell at each of the points where the code could most reasonably be entered without going thru the Command Interpreter's new DOCAT routine. That's exactly what all the JSR's to the routine AllowENTry are doing.

Satisfying that second goal takes up a lot of space, however; and has somewhat undermined the first constraint: WildCAT certainly isn't "in-place" in one place! And I apologize for this rather bizarre, serpentine code; I do hope that now you understand why some things were done the way they were.

Although considerable effort was spent to maintain compatibilty with the existing DOS commands, there were some compromises:

1. While the DOS manual (page 22) states: "To specify drive 1, you use the notation D1 separated from the file name by a comma", you can in fact leave out the comma between CATALOG and D1. With WildCAT that comma is now required; otherwise, it would take the "D" as a filetype and try to find it ... which of course it wouldn't and there would be no files reported. This would also be a problem for Applesoft programs that do something like: PRINT D$"CATALOG D1" without the comma. Therefore, WildCAT issues a (late) "SYNTAX ERROR" message if it encounters an undelimited string of length 2 or more.

2. CATALOG is a favorite routine to execute directly,

APPLIED ENGINEERING INTRODUCES THE TRANSWARP™ ACCELERATOR

The Fastest Apple Accelerator Available

TransWarp is the new accelerator card from Applied Engineering. With a TransWarp card in your Apple II, II+, or IIe all software will run up to 3.6 times faster (3.1 times faster is average). TransWarp works with all Apple software including AppleWorks, Supercalc 3a, Visicalc, and all educational software, graphics and games. TransWarp is compatible with all standard peripheral cards such as Ramworks II and Apple memory cards, Profile and Sider hard disks, 3½" UniDisks, 80 column cards, modems, clock cards, mouses and more. You name it, TransWarp accelerates it! Should 16 bit software become available for the Apple, you can get a low cost 16 bit upgrade chip at any time.

Why TransWarp Is Best

The other speedup cards only speedup your Apple's main memory, but TransWarp has 256K of ultra-fast RAM that accelerates your Apple's main memory, ROM memory and auxiliary memory. And since more and more programs are residing in auxiliary memory, TransWarp can run these programs 3 times faster than the competition. TransWarp doesn't use memory caching, which means TransWarp accelerates all software.

It Couldn't Be Easier

Just plug TransWarp into any slot in your Apple II, II+, or IIe, including slot 3 in the IIe. Just turn on your Apple and zoom... off it runs 3½ times faster. Should you ever want to run at normal speed, just press the ESC key while turning on your Apple. No pre-boot disks are needed. In fact, no software comes with TransWarp because none is needed as the TransWarp acceleration is completely transparent.

All for only $279.

- 3.6 MHZ 65C02
- 256K of ultra-fast on board RAM
- Accelerates main and auxiliary memory
- Low power design for cool operation
- Totally transparent operation with all software
- Plugs into any slot including slot 3 of an Apple IIe
- Accelerated 16 bit option available

TransWarp will make you more productive (3½ times more productive, in fact) because your spreadsheets will recalculate faster, and your word processor will move text in the blink of an eye. Accounting, engineering, educational, tax analysis, and even games will all have warp speed.

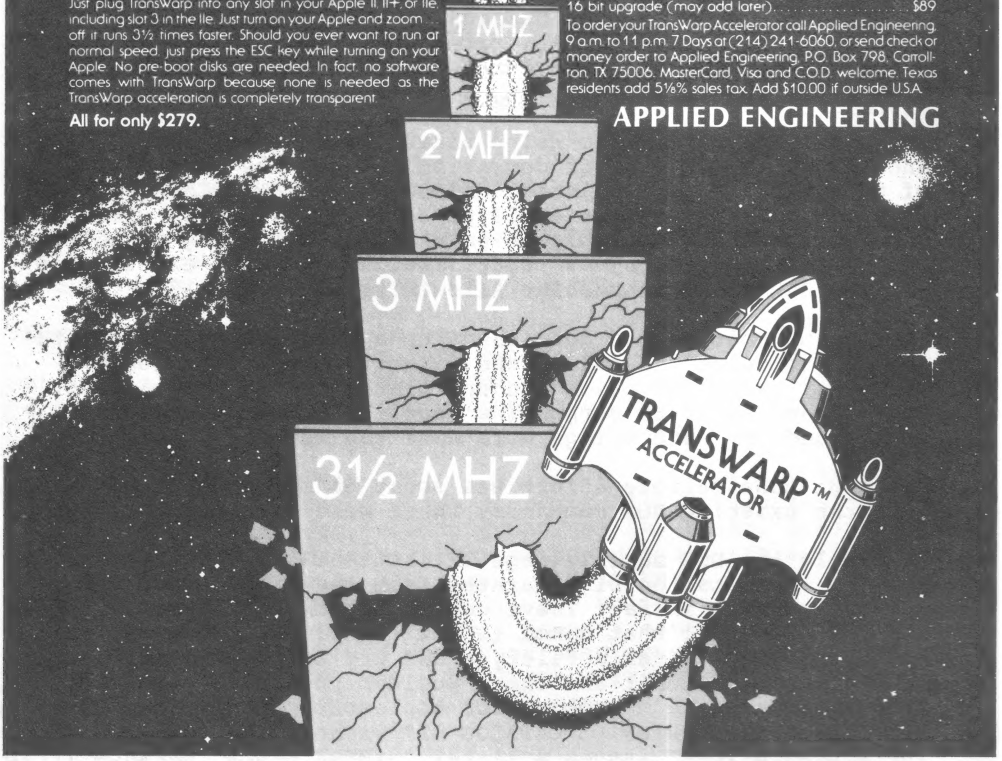
With a TransWarp card in your Apple, you'll easily pass by others using IBM PC's and AT's. In fact, you'll have one of the fastest computers in the galaxy and although TransWarp leaves the competition in the dust, it's still priced far lower, at only $279.

So the next time you're waiting for your computer to finish, give Applied Engineering a call and we'll beam you a TransWarp accelerator card. With our risk free 15 day money back guarantee, you have nothing to lose, except a lot of wasted time.

TransWarp™ Accelerator	$279
16 bit upgrade (may add later)	$89

To order your TransWarp Accelerator call Applied Engineering, 9 a.m. to 11 p.m. 7 Days at (214) 241-6060, or send check or money order to Applied Engineering, P.O. Box 798, Carrollton, TX 75006. MasterCard, Visa and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 if outside U.S.A.

APPLIED ENGINEERING



bypassing the DOS Command Interpreter. FID, for example, provides its CATALOG via the "external" entry to the File Manager, which means that the main entry at CATHNDLR must provide for a "regular" catalog. It is also possible from machine language, however, to bypass both the CI and the FM. This usually involves changing the exit JMP address at DONEXT2 (to return to the user's code) and then jumping directly into almost anywhere in the CATALOG code (see the Listing 1 labels that begin "at"). I believe most of these cases are covered, but you may find some programs, which provide an "internal" CATALOG, that just won't work with WildCAT.

3. In order to both gain some patch space and provide the DIR/CAT short-form command name, the DOS command POSITION was eliminated. You may have to look it up just to find out that it is, much less what it is. Its relative rarity may be due to its implementation: it, like APPEND, finds its way through the file one byte at a time...all day long. Any program that uses it will now get a syntax error. If POSITION is really needed, it can be readily simulated by programming a read-loop to discard N-1 fields before processing the desired Nth field.

The following is a brief commentary on the assembly listing. The paragraph numbers correspond to numbers in comment lines.

The page zero locations I used ($EB thru $EF) are free, i.e. not used by DOS, the Monitor, or the Basics.

(1) In CMDTBL, replace Integer CHAIN address with TYPE and DOCAT address with NewDOCAT.

(2) Rearrange some code (and change both references to it) to add a "print blank" capability. The Command Interpreter uses its own vector to a "COUT" routine via CSW at $36; however, the File Manager (previously) used the Monitor COUT and CROUT routines for printing the catalog. With WildCAT all of DOS now consistently uses the vector at $9FCA for output; plus it has a new BlankOUT routine, all within the original code space.

(3) Recode a very cumbersome form of the "indexed indirect jump" to use register Y and leave X (which is zero by a previous operation) so it can be used in NewDOCAT.

(4) Replace old DOCAT's 12 bytes of code with a JMP to NewDOCAT and use the remainder to space over to column 7 after the file length has been displayed.

(5) NewDOCAT saves the first character of FNAME and substitutes a zero to prevent buffer allocation. It then loads 13, the new Catalog Function Code, and proceeds to CMDHNDLR2. Function 13 enters the catalog code past the "allow for irregular, direct entry".

(6) In the keyword parameter table, change parms to allow a filename with CATALOG and a filename, drive, and slot with DIR. Set new Function 13 address (previously a "no-op" to NOERROR) to WildCAT and change the range check to 14 to allow for it.

(7) Replace the Integer CHAIN code; PrtLOCK displays an asterisk or blank if the file is locked or not.

(8) Shorten the "NO BUFFERS AVAILABLE" message to "NO BUFFER" and re-use the space to decide which Basic is active, then JMP to the appropriate decimal print routine; used to print the free sector value and catalog filesizes. The value to be printed has been previously loaded into A and X.

(9) First, eliminate the need for "NOT DIRECT COMMAND" error message and then re-use the space to check for a "regular" catalog (no filename) or for a catalog by filetype (undelimited, single character). If more than a single, non-blank character is detected (ie, 2nd byte of FNAME is not blank), then "SYNTAX ERROR" message is issued.

(10) At beginning of catalog code allow for most normal points where the code could be directly entered. The new "official" Function 13, WildCAT initializes the FM workarea (per normal) and branches to Read VTOC to "find" the first catalog sector.

(11) Freespace "prolog"; clear carry and branch around another likely "irregular" entry point. Read first/next catalog sector, then lookup and save the filetype. Setup Y with 30 for name length and branch to CkFNAME.

(12) AllowVTOC fakes a "regular" catalog and falls into a JSR to read the VTOC. The BCC to initialize linecount is always taken; only if there had been an I/O error would the carry be set, in which case, control would have passed to the error-message-print exit anyway.

(13) PrtCat displays a catalog line. Note that loc $24, CH, is "POKEd" with 7 for uniform spacing over to the filename. If your printer interface board or 80-column card do not support this convention, then the display will not be properly spaced. The DONEXT routine is unchanged. SKIPLN has been re-arranged to allow init linecount, put out a carriage return, and check for a keypress (Ctrl-Q to quit) after 22 lines. Note: This leaves the cursor in column 37; see below.

(14) CkFNAME "looks under the shell" to figure out what to do. A caret indicates to check for a wildcard string. After JSR to CkCAT, if the equal status is set, then branch to print the catalog line. DoWild checks for the occurence of the wildcard string within the filename. $B4C9,X indexes the name in the Catalog Sector; $AA75,Y indexes the wildcard string; CatNmLen counts from 30 to 0, to scan the whole name.

(15) FreeSpce counts the free sectors, as indicated by the VTOC, loads X and A with the count, and JMPs ToPrtDec.

(16) WaitCk79 provides the "wait" for TYPE; also checks and puts out a carriage return after 79 characters to avoid over-printing long lines on certain printers, such as the MX-80.

(17) TYPE displays the contents of a sequential or random text file. A keypress will pause the display, and Ctrl-Q quits.

(18) InvCOUT is used by both CATALOG and TYPE. It converts hi-bit off characters to proper inverse. It will optionally show control characters as inverse or allow them to "function" as-is; and it will optionally "shift" lower case letters to upper case, if you do not have a lower case adapter; see "...Options" above. Loc $EA, decimal 234, is the Applesoft Hi-Res collision counter; it should always be zero, unless you POKE it.

(19) WaitCQ waits for a keypress and sets the equal status, if Ctrl-Q was pressed.

(20) Replace the inverted phrase DISK VOLUME with FREE SPACE=.

(21) The DOSCMDS list is moved down 6 bytes. AllowENT re-uses these 6 bytes to force a blank in FNamel "under the shell" to ease "irregular" entries into the catalog code; and clears the carry in case the entry was 'atADC9' which also cleared the carry. In the command list, TYPE replaces CHAIN and DIR replaces POSITION; change $A8BF:43 41 D4 to replace with CAT.

(22) Change the two references to DOSCMDS to the new location. These two changes must be done last as the EXEC is changing the very code that is executing.

I would like to thank Lee Reynolds and Art Schumer for their helpful comments and suggestions.

```
                  1000 *SAVE ERV EDGE'S CODE
                  1010 *--------------------------------
EB-               1020 CatLnCnt   .EQ $EB Catalog Linecount
EC-               1030 FType      .EQ $EC Hold looked-up filetype
EE-               1040 FName1     .EQ $EE Hold FNAME shell 1st char
ED-               1050 CatNmLen   .EQ $ED CatName check-length=30
EF-               1060 CatPtLen   .EQ $EF CatName print-length=30
                  1070 *-----(1)----
                  1080          .PH $9D26 In CMDTBL, command addresses,
9D26- B2 B6       1090          .DA TYPE-1 change Integer CHAIN to TYPE
                  1100          .PH $9D3E In CMDTBL, change address to
9D3E- DC A5       1110          .DA NewDOCAT-1 new DOCAT in POSITION code
                  1120 *-----(2)----
                  1130          .PH $9FA8 In ECHO, change old COUT ref
9FA8- CA          1140          .DA #$CA was JSR $9FC5 now JSR $9FCA
                  1150          .PH $9FC5 Cleanup CDI COUT and CROUT
9FC5- A9 A0       1160 BlankOUT LDA #" " and add BLANK out routine
9FC7- 2C          1170          .DA #$2C fake BIT-NOP on fall-thru
9FC8- A9 8D       1180 CROUT    LDA #$8D  DOS vectored CROUT; same loc
9FCA- 6C 36 00    1190 COUT     JMP ($36) DOS vectored COUT; new loc
                  1200          .PH $A710 In PRTERROR, change old COUT
A710- CA          1210          .DA #$CA was JSR $9FC5 now JSR $9FCA
                  1220 *-----(3)----
                  1230          .PH $A186 Cleanup DOCMD; X=0 in NewDOCAT
A186- AC 5F AA    1240          LDY $AA5F CMDINDX
A189- B9 1F 9D    1250          LDA $9D1F,Y CMDTBL+1; use Y instead of X
A18C- 48          1260          PHA
A18D- B9 1E 9D    1270          LDA $9D1E,Y CMDTBL
A190- 48          1280          PHA
A191- 60          1290          RTS
A192- EA          1300          NOP
                  1310 *-----(4)----
                  1320          .PH $A56E Replace old DOCAT code:
A56E- 4C DD A5    1330 OldDOCAT JMP NewDOCAT To allow for direct entry
A571- A5 EC       1340 PrtTYPE LDA FType Print looked-up filetype
A573- 20 CA 9F    1350          JSR COUT and
A576- 4C C5 9F    1360          JMP BlankOUT a blank
A579- EA          1370          NOP
                  1380 *-----(5)----
                  1390          .PH $A5DD Replace old POSITION code:
A5DD- AD 75 AA    1400 NewDOCAT LDA $AA75 FNAME set by CATALOG command
A5E0- 85 EE       1410          STA FName1 save first byte, then zero
A5E2- 8E 75 AA    1420          STX $AA75 to avoid buffer allocation
A5E5- A9 0D       1430          LDA #13 FM WildCAT Function Code
A5E7- 4C AA A2    1440          JMP $A2AA CMDHNDL2 routine, per usual
A5EA- EA          1450          NOP
                  1460 *-----(6)----
                  1470          .PH $A921 DIR [string] [,Dn] [,Sn]
A921- 60 70 00    1480          .DA #$60,$70 ->First comma: is NOT optional
                  1490          .PH $A929 CATALOG [string] <[,Dn] [,Sn]>
A929- 60          1500          .DA #$60 ->Must be CATALOG,D1 or DIR,D2
                  1510          .PH $AAE3 In FM function table, "borrow"
AAE3- 9A AD       1520          .DA WildCAT-1 otherwise useless address
                  1530          .PH $AB10 Change range check from 13 for
AB10- C9 0E       1540          CMP #14 above now USEFULL address
                  1550 *-----(7)----
                  1560          .PH $A4F0 Replace Integer CHAIN code
A4F0- A9 A0       1570 PrtLOCK LDA #" " blank=unlocked
A4F2- BE C8 B4    1580          LDX $B4C8,Y Catalog Filetype entry
A4F5- 10 02       1590          BPL ToPrint
A4F7- A9 AA       1600          LDA #"*" *=locked
A4F9- 4C CA 9F    1610 ToPrint JMP COUT Print " " or "*" indicator
                  1620 *-----(8)----
                  1630          .PH $A9FD Shorten NO BUFFER[S AVAILABLE]
A9FD- D2          1640          .AS -"R" to free 11 bytes for ToPrtDec:
A9FE- 2C 06 E0    1650 ToPrtDec BIT $E006 Check which Basic...
AA01- 30 03       1660          BMI ToInt Integer or
AA03- 4C 24 ED    1670          JMP $ED24 Applesoft; use appropriate
AA06- 4C 1B E5    1680 ToInt JMP $E51B print decimal routine
                  1690 *-----(9)----
                  1700          .PH $A021 Replace JSR ISBASRUN to allow
A021- EA          1710          NOP  ALL commands entered direct
A022- EA          1720          NOP  then error msg is redundant so
A023- EA          1730          NOP  ok to re-use msg space below
```

```
                          .PH $AA2C Replace NOT DIRECT COMMAND msg
AA2C- C9 A0     CkCAT CMP #" " If blank, do regular catalog
AA2E- F0 0C               BEQ ToRTS
AA30- A0 A0               LDY #" " Must be single-char filetype
AA32- CC 76 AA            CPY $AA76 FNAME+1, ie blank afterwards
AA35- F0 03               BEQ CkType if catalog by filetype; else
AA37- 4C C4 A6            JMP $A6C4 CSYNTAX error
AA3A- C5 EC     CkType CMP FType Does filetype match?
AA3C- 60        ToRTS RTS
AA3D- EA                  NOP
AA3E- EA                  NOP
                *----(10)----
                          .PH $AD98
AD98- 20 84 A8  CATHNDLR JSR AllowENT Allow for non-CDI entry
AD9B- 20 DC AB  WildCAT JSR $ABDC Init File Manager Workarea
AD9E- D0 57               BNE ToRWVTOC
ADA0- 4C F4 AD  atADA0 JMP AlowVTOC Allow for non-CDI entry
ADA3- EA        atADA3 NOP  Allow for non-CDI entry and
ADA4- EA                  NOP  alignment
ADA5- 20 84 A8  atADA5 JSR AllowENT Allow for non-CDI entry
ADA8- 20 84 A8  atADA8 JSR AllowENT Allow for non-CDI entry
ADAB- 20 38 AE  atADAB JSR InitCR Init Linecount; output C/R
                *----(11)----
ADAE- 20 2F AE            JSR SKIPLN
ADB1- A2 0C               LDX #12
ADB3- BD AE B3  PrtFreSP LDA FreeMsg-1,X
ADB6- 20 CA 9F            JSR COUT Print " FREE SPACE="
ADB9- CA                  DEX
ADBA- D0 F7               BNE PrtFreSP X=0 for PrtFreSP
ADBC- 20 69 BA            JSR FreeSpce Count & print free sectors
ADBF- 20 2F AE            JSR SKIPLN
ADC2- 20 2F AE            JSR SKIPLN
ADC5- 18                  CLC  Setup for RDNXTDIR to read
ADC6- 90 04               BCC RDNXTDIR first sector; always branch
ADC8- EA                  NOP  alignment
ADC9- 20 84 A8  atADC9 JSR AllowENT Allow non-CDI, non-FM entry
ADCC- 20 11 B0  RDNXTDIR JSR $B011 RDDIRSEC
ADCF- B0 5B               BCS DONEXT2
ADD1- A2 00               LDX #0
ADD3- 8E 9C B3  GTRKNUM STX $B39C DIRINDX
ADD6- BD C6 B4            LDA $B4C6,X Track part of T/S list
ADD9- F0 51               BEQ DONEXT2 If End of Catalog, then exit
ADDB- 30 48               BMI DONEXT If Deleted File, then skip it
ADDD- BD C8 B4            LDA $B4C8,X Catalog Filetype
ADE0- 0A                  ASL  ;skip hi-bit LOCK/UNLOCK flag
ADE1- A0 07               LDY #7
ADE3- 0A        FindTYPE ASL
ADE4- B0 03               BCS GotTYPE
ADE6- 88                  DEY
ADE7- D0 FA               BNE FindTYPE
ADE9- B9 A7 B3  GotTYPE LDA $B3A7,Y From filetype table,
ADEC- 85 EC               STA FType save looked-up filetype
ADEE- A0 1E               LDY #30 Check CatName length and
ADF0- 84 EF               STY CatPtLen Print CatName length
ADF2- D0 4B               BNE CkFNAME always BNE
                *----(12)----
ADF4- 20 84 A8  AlowVTOC JSR AllowENT Allow for non-CDI entry
ADF7- 20 F7 AF  ToRWVTOC JSR $AFF7 RWVTOC read VTOC
ADFA- 90 AF               BCC atADAB always; carry set=I/O ERROR
                *----(13)----
ADFC- EA                  NOP  ;alignment
ADFD- EA                  NOP
ADFE- AC 9C B3  PrtCAT LDY $B39C Restore Y from DIRINDX
AE01- 20 F0 A4            JSR PrtLOCK Print Lock indicator
AE04- 20 71 A5            JSR PrtTYPE Print filetype and BlankOUT
AE07- BE E7 B4            LDX $B4E7,Y Filesize
AE0A- B9 E8 B4            LDA $B4E8,Y Filesize+1
AE0D- 20 FE A9            JSR ToPrtDec Print "true" filesize
AE10- A0 07               LDY #7 "Poke" CH with 7 to "tab"
AE12- 84 24               STY $24 over for filename spacing
AE14- AE 9C B3            LDX $B39C Restore X from DIRINDX
AE17- BD C9 B4  PrtFN LDA $B4C9,X Print Catalog Filename
AE1A- 20 DA B6            JSR InvCOUT with optional conversions
AE1D- E8                  INX
```

```
AE1E- C6 EF     2480          DEC CatPtLen CatName print length
AE20- D0 F5     2490          BNE PrtFN
AE22- 20 2F AE  2500          JSR SKIPLN
AE25- 20 30 B2  2510 DONEXT  JSR $B230 NXTDIREN...atAE25
AE28- 90 A9     2520          BCC GTRKNUM
AE2A- B0 A0     2530          BCS RDNXTDIR
AE2C- 4C 7F B3  2540 DONEXT2 JMP $B37F NOERROR....atAE2C
AE2F- C6 EB     2550 SKIPLN DEC CatLnCnt Linecount..atAE2F
AE31- D0 09     2560          BNE ToCR If not zero, C/R & return
AE33- 20 8D B7  2570          JSR WaitCQ else wait for keypress
AE36- F0 F4     2580          BEQ DONEXT2 If Ctrl-Q, exit to NOERROR
AE38- A9 15     2590 InitCR LDA #22-1 else setup for next 22 lines
AE3A- 85 EB     2600          STA CatLnCnt in line count
AE3C- 4C C8 9F  2610 ToCR JMP CROUT DOS vectored C/R out
                2620 *----(14)----
AE3F- A5 EE     2630 CkFNAME LDA FName1 Holds FNAME first character
AE41- C9 DE     2640          CMP #"^" Wildcard string?
AE43- F0 07     2650          BEQ DoWild yes...maybe
AE45- 20 2C AA  2660          JSR CkCAT Regular or by filetype?
AE48- F0 B4     2670          BEQ PrtCAT yes...else
AE4A- D0 D9     2680          BNE DONEXT none of the above
AE4C- 84 ED     2690 DoWild STY CatNmLen CatName length=30, for NotEQ
AE4E- A0 01     2700          LDY #1 Decr'd to 0; indexes FNAME
AE50- C6 ED     2710 NotEQ DEC CatNmLen Checked all 30 chars?
AE52- 30 D1     2720          BMI DONEXT Yes; no match, do next CatName
AE54- CA        2730 BackDown DEX  Backdown to string match start
AE55- 88        2740          DEY  Backdown to 0, ie. FNAME start
AE56- D0 FC     2750          BNE BackDown
AE58- C8        2760 YesEQ INY  First Y=1, then on past "^"
AE59- E8        2770          INX
AE5A- B9 75 AA  2780          LDA $AA75,Y FNAME
AE5D- C9 A0     2790          CMP #" " If blank then wildcard EOS and
AE5F- F0 9D     2800          BEQ PrtCAT still =, so we have a match!
AE61- DD C9 B4  2810          CMP $B4C9,X FNAME = CatName?
AE64- F0 F2     2820          BEQ YesEQ
AE66- E8        2830          INX  No, setup X to backdown 1 past
AE67- D0 E7     2840          BNE NotEQ string match start; always BNE
AE69- EA        2850          NOP
                2860 *----(15)----
                2870          .PH $BA69 Catalog Free Space Patch
BA69- 86 44     2880 FreeSpce STX $44 X=0
BA6B- 86 45     2890          STX $45 Init Free Sec Count var
BA6D- A0 C8     2900          LDY #50*4 VTOC entries * entry length
BA6F- B9 F2 B3  2910 NxBitMap LDA $B3F2,Y BITMAP-1 in VTOC buffer
BA72- 0A        2920 CkFree ASL  ;shift hi-order bit into CARRY
BA73- 90 06     2930          BCC CkMore In use, so check if any more
BA75- E6 44     2940          INC $44 Incr free sector count
BA77- D0 F9     2950          BNE CkFree Zero means > 255, so
BA79- E6 45     2960          INC $45 incr "page" part of word
BA7B- D0 F5     2970 CkMore BNE CkFree More bits in same byte?
BA7D- 88        2980          DEY  decr index to next VTOC byte
BA7E- D0 EF     2990          BNE NxBitMap All done?
BA80- A6 44     3000          LDX $44 Yes, so setup count in X & A
BA82- A5 45     3010          LDA $45 for decimal print via
BA84- 4C FE A9  3020          JMP ToPrtDec one of the BASICs
                3030 *----(16)----
BA87- 20 A8 FC  3040 WaitCk79 JSR $FCA8 Monitor WAIT routine
BA8A- C6 55     3050          DEC $55 Decr char cnt
BA8C- D0 82     3060          BNE $BA10 Fortuitous RTS; else fall thru
BA8E- A9 4F     3070 InitLine LDA #79 TYPE prolog/setup
BA90- 85 55     3080          STA $55 Init printer 80-col char cnt
BA92- 4C C8 9F  3090          JMP CROUT
BA95- EA        3100          NOP
                3110 *----(17)----
                3120          .PH $B6B3
B6B3- 20 A3 A2  3130 TYPE JSR $A2A3 DOS Open file
B6B6- 20 8E BA  3140 DoInitLn JSR InitLine Init char cnt & CROUT
B6B9- 20 8C A6  3150 ToRead JSR $A68C DOS Read char
B6BC- F0 14     3160          BEQ ToWaitCQ EOF maybe...Ctrl-Q quit?
B6BE- C9 8D     3170          CMP #$8D Carriage return?
B6C0- F0 F4     3180          BEQ DoInitLn Yes, handle immediately
B6C2- 20 DA B6  3190          JSR InvCOUT Optional Ctrls & Hibit=0 INV
B6C5- A5 F1     3200          LDA $F1 Applesoft SPEED=nn byte
B6C7- 20 87 BA  3210          JSR WaitCk79 Wait SPEED; 79 chars yet?
B6CA- AD 00 C0  3220          LDA $C000 Has a key been pressed?
B6CD- 10 EA     3230          BPL ToRead No, read on
B6CF- 8D 10 C0  3240          STA $C010 Reset keyboard strobe
```

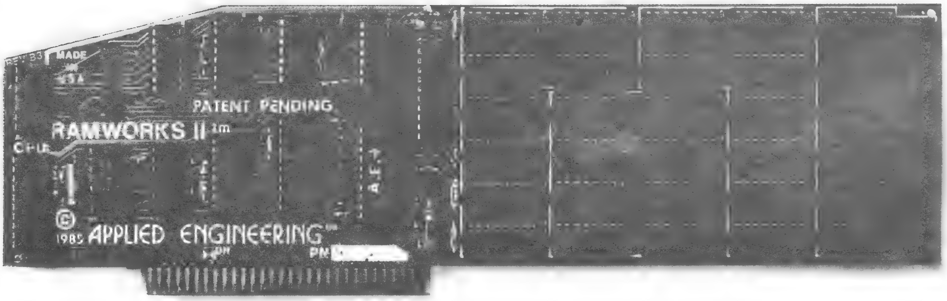
```
B6D2- 20 8D B7 3250 ToWaitCQ JSR WaitCQ Wait keypress, check Ctrl-Q?
B6D5- D0 E2    3260          BNE ToRead If not Ctrl-Q, read on
B6D7- 4C FC A2 3270          JMP $A2FC DOS Close, Deallocate, Exit
               3280 *----(18)----
B6DA- A8       3290 InvCOUT TAY  If < $80, then hibit off
B6DB- 10 08    3300          BPL SetINV so set inverse flag & convert
B6DD- C9 A0    3310          CMP #$A0 Ctrl-char?
B6DF- B0 0E    3320          BCS CkLoCase No
B6E1- 24 EA    3330          BIT $EA Usually, loc 234 contains 0:
B6E3- 30 0A    3340          BMI CkLoCase POKE 234,255 skips conversion
B6E5- 46 32    3350 SetINV LSR $32 Set Inverse by shifting 0 into
B6E7- 46 32    3360          LSR $32 INVFLG first 2 bits; set carry
B6E9- 29 3F    3370          AND #$3F Turn off 1st 2 bits maps down
B6EB- 69 1F    3380          ADC #$1F maps up into hibit-on part of
B6ED- 49 E0    3390          EOR #$E0 upper-case screen-char range
B6EF- C9 E0    3400 CkLoCase CMP #$E0 Lower-case?
B6F1- 90 02    3410          BCC ToCOUT No; but POKE -18700,223 or
B6F3- 29 FF    3420          AND #$FF B6F4:DF shifts l.c. to U.C.
B6F5- 20 CA 9F 3430 ToCOUT JSR COUT DOS vectored COUT
B6F8- A9 FF    3440          LDA #$FF
B6FA- 85 32    3450          STA $32 Set normal video; always
B6FC- 60       3460          RTS
               3470 *----(19)----
               3480          .PH $B78D Wait keypress; check Ctrl-Q
B78D- 20 0C FD 3490 WaitCQ JSR $FD0C Monitor RDKEY
B790- C9 91    3500          CMP #$91 Was it Ctrl-Q?
B792- 60       3510          RTS
               3520 *----(20)----
               3530          .PH $B3AF Replace: DISK VOLUME inverted
B3AF- BD C5 C3
B3B2- C1 D0 D3
B3B5- A0 C5 C5
B3B8- D2 C6 A0 3540 FreeMsg .AS -"=ECAPS EERF " with FREE SPACE=
               3550 *----(21)----
               3560          .PH $A884 Setup FName1 for "irregular"
A884- A9 A0    3570 AllowENT LDA #" " entry into CATALOG code
A886- 85 EE    3580          STA FName1 Force blank at CkFNAME above
A888- 18       3590          CLC  For possible RDNXTDIR entry
A889- 60       3600          RTS
A88A- 49 4E 49
A88D- D4       3610 DOSCMDS .AT 'INIT' Move down DOSCMDS table and
A88E- 4C 4F 41
A891- C4       3620          .AT 'LOAD' re-use the freed space above
A892- 53 41 56
A895- C5       3630          .AT 'SAVE'
A896- 52 55 CE 3640          .AT 'RUN'
A899- 54 59 50
A89C- C5       3650          .AT 'TYPE' was CHAIN
A89D- 44 45 4C
A8A0- 45 54 C5 3660          .AT 'DELETE'
A8A3- 4C 4F 43
A8A6- CB       3670          .AT 'LOCK'
A8A7- 55 4E 4C
A8AA- 4F 43 CB 3680          .AT 'UNLOCK'
A8AD- 43 4C 4F
A8B0- 53 C5    3690          .AT 'CLOSE'
A8B2- 52 45 41
A8B5- C4       3700          .AT 'READ'
A8B6- 45 58 45
A8B9- C3       3710          .AT 'EXEC'
A8BA- 57 52 49
A8BD- 54 C5    3720          .AT 'WRITE'
A8BF- 44 49 D2 3730 atA8BF .AT 'DIR' was POSITION; for CAT:43 41 D4
               3740 *----(22)----
               3750          .PH $9FFB In Command Interpreter PARSE
9FFB- B9 8A A8 3760          LDA DOSCMDS,Y DOSCMDS table ref. was $A884
               3770          .PH $9FED Only 2 references to DOSCMDS
9FED- 59 8A A8 3780          EOR DOSCMDS,Y DO THIS AFTER ABOVE CHANGES!
               3790 *------------------------------
```

```
CALL -151
9D26:B2 B6
9D3E:DC A5
9FA8:CA
9FC5:A9 A0 2C A9 8D 6C 36 00
A710:CA
A186:AC 5F AA B9 1F 9D 48 B9
    :1E 9D 48 60 EA
A56E:4C DD A5 A5 EC 20 CA 9F
    :4C C5 9F EA
A5DD:AD 75 AA 85 EE 8E 75 AA
    :A9 0D 4C AA A2 EA
A921:60 70
A929:60
AAE3:9A AD
AB10:C9 0E
A4F0:A9 A0 BE C8 B4 10 02 A9
    :AA 4C CA 9F
A9FD:D2 2C 06 E0 30 03 4C 24
    :ED 4C 1B E5
A021:EA EA EA
AA2C:C9 A0 F0 0C A0 A0 CC 76
    :AA F0 03 4C C4 A6 C5 EC
    :60 EA EA
AD98:20 84 A8 20 DC AB D0 57
    :4C F4 AD EA EA 20 84 A8
    :20 84 A8 20 38 AE
ADAE:20 2F AE A2 0C BD AE B3
    :20 CA 9F CA D0 F7 20 69
    :BA 20 2F AE 20 2F AE
ADC5:18 90 04 EA 20 84 A8 20
    :11 B0 B0 5B A2 00 8E 9C
    :B3 BD C6 B4 F0 51 30 48
ADDD:BD C8 B4 0A A0 07 0A B0
    :03 88 D0 FA
ADE9:B9 A7 B3 85 EC A0 1E 84
    :EF D0 4B
ADF4:20 84 A8 20 F7 AF 90 AF
ADFC:EA EA AC 9C B3 20 F0 A4
    :20 71 A5
AE07:BE E7 B4 B9 E8 B4 20 FE
    :A9 A0 07 84 24 AE 9C B3
AE17:BD C9 B4 20 DA B6 E8 C6
    :EF D0 F5 20 2F AE
```

```
AE25:20 30 B2 90 A9 B0 A0 4C
    :7F B3
AE2F:C6 EB D0 09 20 8D B7 F0
    :F4 A9 15 85 EB 4C C8 9F
AE3F:A5 EE C9 DE F0 07 20 2C
    :AA F0 B4 D0 D9
AE4C:84 ED A0 01 C6 ED 30 D1
    :CA 88 D0 FC C8 E8 B9 75
    :AA C9 A0 F0 9D DD C9 B4
    :F0 F2 E8 D0 E7 EA
BA69:86 44 86 45 A0 C8 B9 F2
    :B3 0A 90 06 E6 44 D0 F9
    :E6 45 D0 F5 88 D0 EF A6
    :44 A5 45 4C FE A9
BA87:20 A8 FC C6 55 D0 82 A9
    :4F 85 55 4C C8 9F EA
B6B3:20 A3 A2 20 8E BA 20 8C
    :A6 F0 14 C9 8D F0 F4 20
    :DA B6 A5 F1 20 87 BA AD
    :00 C0 10 EA 8D 10 C0 20
    :8D B7 D0 E2 4C FC A2
B6DA:A8 10 08 C9 A0 B0 0E 24
    :EA 30 0A 46 32 46 32 29
    :3F 69 1F 49 E0
B6EF:C9 E0 90 02 29 FF 20 CA
    :9F A9 FF 85 32 60
B78D:20 0C FD C9 91 60
B3AF:BD C5 C3 C1 D0 D3 A0 C5
    :C5 D2 C6 A0
A884:A9 A0 85 EE 18 60
A88A:49 4E 49 D4 4C 4F 41 C4
    :53 41 56 C5 52 55 CE 54
    :59 50 C5
A89D:44 45 4C 45 54 C5 4C 4F
    :43 CB 55 4E 4C 4F 43 CB
A8AD:43 4C 4F 53 C5 52 45 41
    :C4 45 58 45 C3 57 52 49
    :54 C5
A8BF:44 49 D2
9FFB:B9 8A A8
9FED:59 8A A8
48:04 N 3D0G
```

Meet RamWorks II®

The Recognized Industry Standard For Memory Expansion of the Apple IIe.



RamWorks II. A Generation Ahead. Again.

The best selling expansion card for the Apple IIe just got even better. With RamWorks II, expand your IIe to an incredible 3 megabytes of usable RAM.

Turbo Charged AppleWorks.

RamWorks II plugs into the IIe auxiliary slot and acts just like Apple's extended 80 column card, only better— because if you buy a 256K or larger card, AppleWorks will automatically load itself into RamWorks II. This dramatically increases AppleWorks' speed and power because it effectively eliminates the time required to access disk drive 1. Now, switch from word processing to spreadsheet to database management at the speed of light. AppleWorks responds the moment your fingers touch the keyboard.

But AppleWorks has certain internal limits, independent of available memory. Fear not. Only RamWorks II (and the original RamWorks of course) removes those limits. Only RamWorks II increases the maximum number of records available from 1,350 to over 16,000. Only RamWorks II actually increases the number of lines permitted in the word processing mode. And only RamWorks II features a built-in printer buffer, so you no longer have to wait for your printer to stop before going back to AppleWorks (256K or larger RamWorks II required).

With RamWorks II, you won't have to split your data into 2 or more separate files because you'll have the necessary memory to access ALL your data ALL the time, quickly and conveniently.

RamWorks II	AppleWorks Desktop
128K	101K
256K	188K
512K	378K
1 MEG	758K
1.5 MEG	1136K
3 MEG	2277K

The Most Friendly, Most Expandable Card Available.

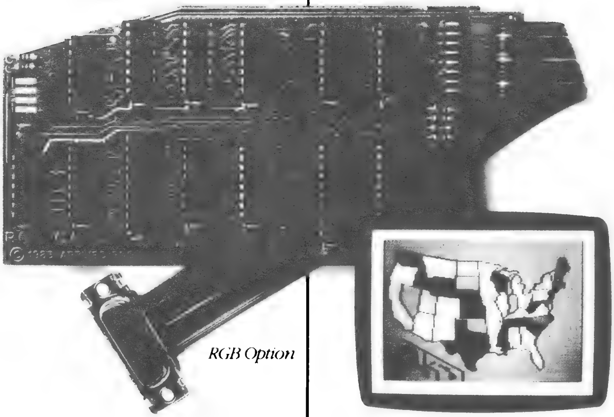
RamWorks II is compatible with more off-the-shelf software than any other RAM card. Popular programs like Advanced VisiCalc, Magic Office System, Flashcalc, The Spread Sheet, Diversi-DOS, Supercalc 3A, Magicalc, etc. (and hardware add-ons like Profile and Sider hard disks). Fact is, only RamWorks is 100% compatible with all software written for the Apple 80 column and extended 80 column cards. In addition, RamWorks II can emulate most other RAM cards, so you can use programs written for them without modification. And any size RamWorks II can be user upgraded later to any larger size.

RamWorks II was designed so you could take full advantage of future developments in 16 and 32 bit microprocessors. As your needs grow, so can RamWorks II. A handy coprocessor connector allows the latest and greatest coprocessor cards to access all 3 MEG of RamWorks II memory. And speaking of more memory, RamWorks II has a memory expansion connector on board so a low profile (no slot 1 interference) memory expansion card can add another 512K or 2 MEG of memory.

Unlike Apple's smaller, more expensive RAM card, RamWorks II plugs into the IIe auxiliary slot and therefore leaves slots 4 and 5 available for other peripheral cards.

It's In Color

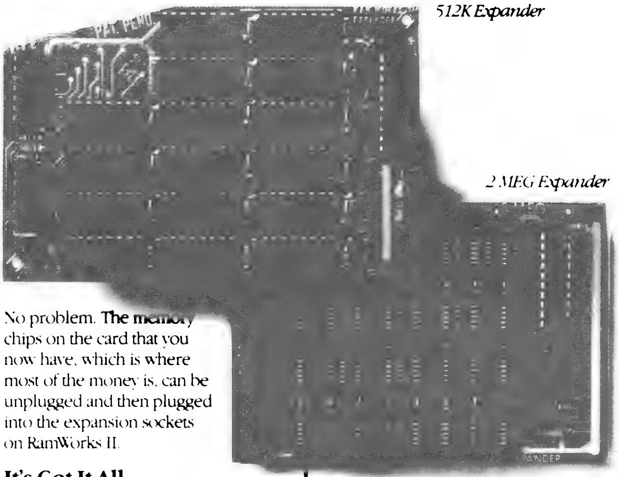
RamWorks II by itself is *fully* compatible with both the Apple monochrome and color monitors. But if you want better color graphics *plus* a more readable 80 column text (that blows away any composite color monitor) you'll appreciate our RGB color option. For only $129, it can be added to RamWorks II, giving you a razor sharp, vivid brilliance that's unsurpassed in the industry. The RGB option does not waste another valuable slot, but rather plugs into the back of RamWorks II with no slot 1 interference (works on the original RamWorks, too) and attaches to virtually any RGB monitor. And remember: You can order the RGB option with your RamWorks II. Or add it on at a later date.



RGB Option

It Corrects Mistakes.

Let's say you bought some other RAM card (and that's a mistake) and your RAM card is not being recognized by AppleWorks, Advanced Visicalc, Flashcalc, Supercalc 3A, or other programs, and you want RamWorks II.



512K Expander

2 MEG Expander

No problem. The memory chips on the card that you now have, which is where most of the money is, can be unplugged and then plugged into the expansion sockets on RamWorks II.

It's Got It All.

- 15 Day Money Back Guarantee
- Super sharp 80 column text (patent pending) with or without RGB option
- Double high resolution graphics (with or without RGB option)
- Expandable up to 1 Meg (1024K) on main board
- Expandable to 3 Meg (3072K) with expander (piggyback) card
- Can use 64K or 256K RAMS in any combination
- Linear addressing coprocessor port
- Automatic AppleWorks expansion up to 2277K desktop
- Accelerates AppleWorks
- Built-in AppleWorks printer buffer
- The only large RAM card that's 100% compatible with all IIe software
- RamDrive ™ the ultimate disk emulation software included free
- 16 Bit option
- Compatible RGB option
- Built-in self diagnostics software
- No slot 1 interference
- Lowest power consumption (patent pending)
- Takes only one slot (auxiliary)
- Software industry standard
- Advanced Computer Aided Design
- Used by Apple Computer, Steve Wozniak and virtually all software companies
- Displays date and time on the AppleWorks screen with any PRO-DOS compatible clock
- 5 Year no hassle warranty

RamWorks II with 64K	**$ 179**
RamWorks II with 256K	**$ 219**
RamWorks II with 512K	**$ 269**
RamWorks II with 1 MEG	**$ 389**
RamWorks II with 1.5 MEG	**$ 549**
RamWorks II with 3 MEG	**$1699**
RGB Option (may add later)	**$ 129**
16 Bit Option (may add later)	**$ 89**

RamWorks II. The industry standard for memory expansion of the Apple IIe.

ORDER YOUR RamWorks II TODAY. 9 a.m. to 11 p.m. 7 Days, or send check or money order to Applied Engineering. MasterCard, Visa and C.O.D. welcome. Texas Residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Mitsubishi 50740 Series

I received information from several sources this week about an interesting new branch of the 6502 tree. Mitsubishi has published specs for eight varieties, all part of the "740 series". The chip is based on the 6502, adds some new addressing modes for some of the standard 6502 opcodes, and adds 13 new opcodes. (Unfortunately, the opcode enhancements are not compatible with any of the other enhanced 6502s.)

The chips in the 740 series are intended for use as microcontrollers. As such, most of them have on-chip RAM and ROM. They all have built-in I/O ports, timers, and other goodies. The most interesting (to Don Lancaster, Nigel Nathan, and me) is the M50/34. This chip, said to cost only $12, has four A/D converters, UART, six timers, a serial I/O port, four 8-bit I/O ports, a pair of stepper-motor drivers, and more. It all lives in a 64-pin shrink-DIP package. The M50734 is the only one in the 740 series which has no internal ROM and RAM. It is CMOS. The clock runs at 8 MHz, which in effect runs the opcodes at 2 MHz (that is, two cycle instructions take one microsecond).

To control all these functions, the bytes in page zero from $DA through $FF are used as I/O, control, and status registers.

One of the trickiest enhancements allows direct access (without bank switching or bank registers) to a second 64K memory, for data only. Apparently one of the address modes changes the state of one of the output signals during data memory references; if you use that signal to enable another bank of memory. ALMOST like having direct-addressability of 128K.

The data bus is multiplexed with half of the address bus, so it's a little harder to interface. Naturally, to get all the functions I mentioned above with only 64 pins, there have to be shared pins. Depending on which functions you are using, some of the timers and some of the I/O pins have dedicated uses.

The 6502 has one unused status bit. The 740 series calls this the T-flag, and gives it a use. If T=1, a special address mode is enabled which allows memory-to-memory operations without using the A-register. As I understand it, when T=1, address modes which use X as an index register take on a new meaning: rather than moving data between the indexed address and the A-register, data is moved between the absolute address and the zero page location whose address is in the X-register. If I am correct, ADC $400,X (assume X contains $34) would add the contents of $400 to the contents of $34, and store the result in $34. If T=0, indexing works in the old-fashioned 6502 way.

Another powerful enhancement allows you call subroutines with a two-byte version of the JSR opcode. One variation uses vectors stored in page zero, and the other uses vectors stored $FF00 through $FFF3. JMP can also uses vectors stored in page zero, so you have a two-byte JMP indirect.

Four new opcodes give you the ability to set, clear, or test any bit in the A-register or in page zero. This uses up 64 opcodes, because the bit number and bit state are coded into the opcode byte. Rockwell's version of the 65C02 includes page-zero bit-addressing, but the opcodes are not the same.

There are other new instructions, including several about which I do not have accurate complete information.

```
      RRF zp      (I think it swaps nybbles in the byte)
      COM zp      (Probably forms 2's complement at zp)
      LDM zp      (Probably loads ABS(zp) into A-register)
      CLT         clear T-bit in status
      SET         set T-bit in status
      STP         stop the clock until reset or interrupt
      WIT         low power mode   "     "   "      "
      SLW         (slow?)
      FST         (fast?)
      INC         increment A-reg
      DEC         decrement A-reg
      BRA rel     branch always.
```

Of all the extensions, only ONE (BRA) is compatible with the standard 65C02 and 65816 extensions from Western Design Center (the OFFICIAL source for 6502 designs). The others, even if they do the same thing, use a different opcode value. Why?

If you have worked up an appetite for more information on the 740 series, contact Mitsubishi. I don't have all their numbers, but you can get close by calling 1-800-421-1132.

When we get all the data, we will be writing a Cross Assembler so you can use your Apple to develop software for this chip.

Faster Cyclic Redundancy Checking.........Bob Sander-Cederlof

In the April 1984 issue of AAL I showed how to compute a cyclic redundancy check code (CRC) for a buffer full of data. I also tried to explain a little of the theory, as much as I understood. In the June 1984 issue Bruce Love explained how to work backward from the computed CRC of a received buffer to correct a single bit error. Both of these programs were written in plain 6502 code.

In the February 1986 "Dr. Dobb's Journal" Terry Ritter writes about "The Great CRC Mystery". He also presents some Pascal programs and 8088 machine code programs for calculating the CRC in various ways. Terry describes very briefly a table driven method (the very fastest way) and a byte-oriented method (almost as fast as table-driven).

I translated Terry's machine-coded byte-oriented method from 8088 to 65802 code, but even after twiddling and tweaking for half a day I could not make it give the correct answers. I don't know if his method is correct or not, but of course it MUST be, since it is printed in Dr. Dobb's and since he claims it works and since he even tells how many milliseconds it takes.

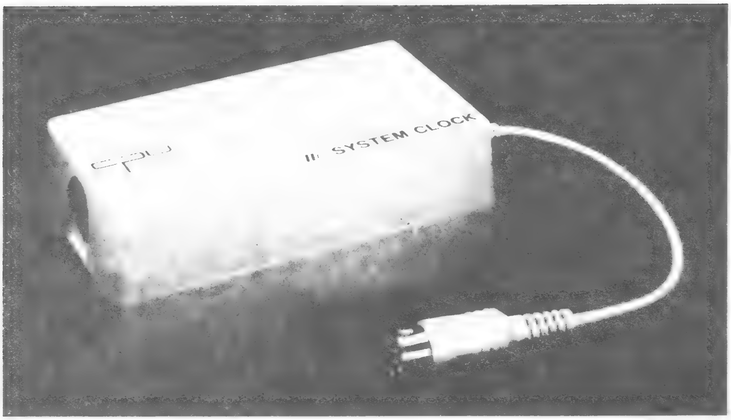
Anyway, I decided to derive my own byte-oriented method. The CRC algorithm is basically a "long division" of the entire bit stream in the buffer as though it were one long binary word. The divisor is $11021 in the CCITT scheme. The check code we use is the remainder of the division. The normal algorithm does "long division" on a bit-by-bit basis. The byte-oriented algorithm does "long division" on a byte-by-byte basis.

I put long division in quotation marks above because it is not EXACTLY long division. The difference is that the subtraction steps are replaced with exclusive-or operations. The exclusive-or is performed whenever the leading bit of the new dividend is a 1-bit. Here is a fully worked out example, for a CRC-so-far = $E1F0, and the next byte = $CC:

```
       "divide"   $E1F0CC by $11021, "quotient bits" down
       the left edge.  Next CRC is the "remainder"

       1110 0001 1111 0000 1100 1100   (E1F0CC in binary)
1  eor 1000 1000 0001 0000 1           (11021 in binary)
       ---------------------
        110 1001 1110 0000 01
1   eor 100 0100 0000 1000 01
        ---------------------
         10 1101 1110 1000 000
1    eor 10 0010 0000 0100 001
         ---------------------
0         0 1111 1110 1100 0010
            1111 1110 1100 0010 1
1     eor   1000 1000 0001 0000 1
            ---------------------
             111 0110 1101 0010 01
1      eor   100 0100 0000 1000 01
```

NEW PRODUCT



IIc SYSTEM CLOCK

- Fully ProDos compatible
- Automatic time and date stamping
- Easy to use from BASIC
- Battery operated, uses 3 "AA" batteries (will last 1-2 years before simple replacement)
- Date has year, month, date and day of week
- Time has hours, minutes and seconds
- Will time and date stamp AppleWorks files
- Will display time and date on the AppleWorks screen
- Auto access from AppleWorks data-base (just use a time and date field)
- Pass through serial port - The IIc system clock can plug into either the modem or printer serial port, then modem or printer plugs into the clock
- No hassle 5 year warranty
- Only $79.00

ÆE

"We Set the Standard"

214-241-6060 **APPLIED ENGINEERING** 9 AM - 11 PM

```
             ---------------------
              11 0010 1101 1010 000
1       eor   10 0010 0000 0100 001
               ---------------------
               1 0000 1101 1110 0010
1        eor   1 0001 0000 0010 0001
               ---------------------
                 0001 1101 1100 0011 = $1DC3
```

Note that the "quotient" is $EF. This "quotient" can always be exactly computed by using just the first byte of the dividend (the high byte of the old CRC code): quotient = crchi .eor. crchi/16. If you carefully study the worked out example above, you should be able to see why this is true. Now, if we use the exclusive-or rather than addition to perform a multiplication of the quotient times $11021, it will look like this:

```
                      uuuu.vvvv  (symbolic quotient in binary)
                       x $11021  (multiplier in hexadecimal)
                       --------
                      uuuu.vvvv
               u.uuuu.vvv0
       uuuu.vvvv
  uuuu.vvvv
  -----------------------------
    whatever........
```

There are several significant things to notice about the multiplication above. First, we only need to save the rightmost 16 bits of the "product". If we exclusive-or those bits with the rightmost 16 bits of the original dividend (which means the low byte of the old CRC followed by the new byte), we will get the next CRC. (This trick relies on the fact that exclusive-or is a reversible operation, so that "adding" and "subtracting" give the same result!)

Furthermore, we can organize those "partial products" in a more efficient way for computation. Now, let's write the original CRC symbolically as "aaaa.bbbb.cccc.dddd", and the next data byte as "eeee.ffff". The "quotient" after "dividing" by $11021 will be "aaaa.bbbb exclusive-or 0000.aaaa"; let's write that symbolically as "aaaa.gggg". Then we can compute the next CRC code by the following very simple steps:

```
       cccc.dddd.eeee.ffff
  eor  gggg.0000.aaaa.gggg
  eor  000a.aaag.ggg0.0000
       -------------------
       wwww.xxxx.yyyy.zzzz
```

Believe it or not!

The program that follows implements this algorithm, in lines 1550-1760. I used 65802 code, but it really could be done quite nicely in plain 6502 as well. I leave it as "an exercise for the reader" (as college textbooks are wont to say), should you wish to try the algorithm in a plain-vanilla 6502.

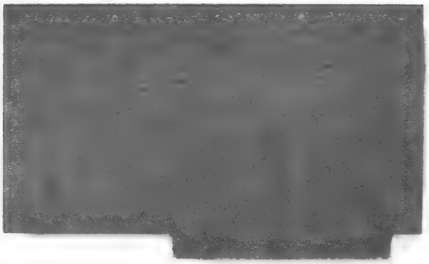
The SEND and RECV programs simulate sending and receiving a buffer-full of data. I chose to put my buffer at $4000, for 258 bytes. This is the same as in the April 1984 article.

The FIND.BAD.BIT program is simply a translation of Bruce Love's 1984 program into 65802 code. Thanks to 16-bit registers, it is significantly faster and shorter.

Speaking of speed, the code for computing the next CRC code for one new byte takes (if I counted correctly) 57 clock cycles. In a normal Apple that means about 56 microseconds. The time for 8088 machine code in Terry Ritter's article was 17 microseconds for the equivalent steps. He was running with a 7.16 MHz clock. If you ran the 65802 code in an Applied Engineering Transwarp card or a Titan Accelerator card with a 4-MHz 65802 (running at 3.58 MHz), the time would be only 15.9 microseconds in an Apple.

```
                  1000 *SAVE S.CRC GENERATOR
                  1010 *--------------------------------
4000-             1020 BUFFER .EQ $4000
4102-             1030 LIMIT  .EQ $4102
                  1040 *--------------------------------
00-               1050 CRC    .EQ $00,01
02-               1060 PNTR   .EQ $02,03
0A-               1070 TEMP   .EQ $0A,0B
                  1080 *--------------------------------
F941-             1090 PRNTAX .EQ $F941
FD8E-             1100 CROUT  .EQ $FD8E
                  1110 *--------------------------------
                  1120 *      SIMULATE SENDING A BUFFER-FULL
                  1130 *--------------------------------
                  1140 SEND
0800- A9 00       1150        LDA #0       CLEAR CRC BYTES IN BUFFER
0802- 8D 01 41    1160        STA LIMIT-1
0805- 8D 00 41    1170        STA LIMIT-2
0808- 20 28 08    1180        JSR NEW.CRC.BUFFER    COMPUTE CRC OF 258 BYTES
080B- A6 00       1190        LDX CRC      STORE CRC INTO LAST 2 BYTES
080D- A5 01       1200        LDA CRC+1
080F- 8E 01 41    1210        STX LIMIT-1
0812- 8D 00 41    1220        STA LIMIT-2
0815- 20 41 F9    1230        JSR PRNTAX   DISPLAY THE CRC
0818- 4C 8E FD    1240        JMP CROUT    <RETURN> AND RETURN
                  1250 *--------------------------------
                  1260 *      SIMULATE RECEIVING A BUFFER-FULL
                  1270 *--------------------------------
                  1280 RECV
081B- 20 28 08    1290        JSR NEW.CRC.BUFFER    COMPUTE CRC OF 258 BYTES
081E- A6 00       1300        LDX CRC      DISPLAY CRC IN HEX
0820- A5 01       1310        LDA CRC+1
0822- 20 41 F9    1320        JSR PRNTAX
0825- 4C 8E FD    1330        JMP CROUT
                  1340 *--------------------------------
                       1350        .OP 65802
                       1360 *--------------------------------
                       1370 *      CRCH      CRCL      DATA
                       1380 *   aaaa.bbbb.cccc.dddd.eeee.ffff
                       1390 *  +0000.aaaa
                       1400 *   ---------
                       1410 *   aaaa.gggg
                       1420 *           +gggg.0000.aaaa.gggg
                       1430 *           +000a.aaag.ggg0.0000
                       1440 *            -------------------
                       1450 *            (crchi)   (crclo)
                       1460 *--------------------------------
```

With Z-80 Plus,™ run CP/M®–the largest body of software in existence.



Now, get two computers in one, and all the advantages of both.

Enter the CP/M world with the new Z-80 Plus card from Applied Engineering, and introduce your Apple IIe® or II + ® to the thousands of CP/M programs. Only the Z-80 Plus comes standard with the new 4.0 software, the most advanced system ever for running CP/M programs

The new 4.0 boasts advanced features like built-in disk emulation for popular memory expansion boards, boosting both system speed and storage capacity And menu-driven utilities that let you get to work faster. The Z-80 Plus also lets you run older CP/M programs — all the way down to Version 1.6 (2.2 is the most popular).

The Z-80 Plus is the only card on the market capable of accessing more than 64K in an Apple IIe. If you have an extended 80-column card, all 128K is usable, and if you have RamWorks, up to 1088K is available.

Each Z-80 Plus comes with our CP/M Ram Drive software, enabling IIe owners to use an extended 80-column card or a RamWorks card as a high-speed Ram disk which runs CP/M software up to *twenty times faster* So packages like WordStar and dBASE II run at blinding speed.

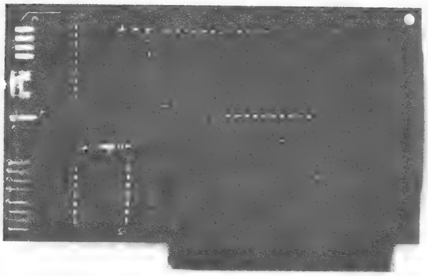
Simply plug the Z-80 Plus into any slot in your Apple. You'll get the benefits of two computers in one — all at an unbelievably low price (only $139!).

• Fully compatible with ALL CP/M software • Fully compatible with most hard disks, including Corvus and the Sider • Fully compatible with Microsoft disks (no pre-boot required) • Specifically designed for high speed operation in the Apple IIe (runs just as fast in the Apple II + and Franklin) • Runs WordStar, dBASE II, Turbo Pascal, Fortran-80, Peachtree and ALL other CP/M software with *no pre-boot* • Semi-custom I.C. and low parts count allows Z-80 Plus to fly through CP/M programs with extremely low power consumption (we use the Z-80B) • Does EVERYTHING other Z-80 boards do, *plus* Z-80 interrupts • Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Æ Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Timemaster H.O.,™ the only clock that displays time and date on AppleWorks'™ screens and files.



Now, get all the features of all the competition combined!

It's the smart way to put the time and date on your Apple II + ® or IIe®. Because only the Timemaster H.O. packs *ALL* the features of all the competition *combined*, including leap year, year (not just in PRO-DOS), month, date, day of week, hours, minutes, seconds and milliseconds. It's totally PRO-DOS, DOS 3.3, PASCAL and CP/M compatible. And of course, it works better than any other clock with AppleWorks.

If you're using or writing software for other clock cards, you're still covered. Because the H.O. will *automatically* emulate them. And the Timemaster H.O. adds 14 new commands to BASIC. The H.O. even comes complete with two disks full of sample programs, including a computerized appointment book, a DOS dating program, interrupt programs, and over 30 programs that others charge extra for — or don't even offer

As a low-cost option, you can add true BSR remote control to the H.O., giving you remote control of up to 16 lights and appliances in your home or office.

• Fully PRO-DOS and DOS 3.3, CP/M and PASCAL compatible • Time in hours, minutes, seconds and milliseconds (the ONLY PRO-DOS compatible card with millisecond capability); date with year, month, day of week and leap year • 24-Hour military format or 12-hour AM/PM format • Eight software controlled interrupts so you can run two programs at the same time (many examples included) • Allows AppleWorks to time and date stamp all data automatically • The only clock card that displays time and date on the AppleWorks screen • Five year warranty

Clock price	$129.00
BSR option (may be added later)	$ 49.00

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Æ Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

```
                          1470 NEW.CRC.BUFFER
000828- 18                1480          CLC
000829- FB                1490          XCE
00082A- C2 30             1500          REP #$30     M&X BOTH 16-BITS
00082C- A9 FF FF          1510          LDA ##$FFFF
00082F- 85 00             1520          STA CRC      INITIALIZE CRC FOR BUFFER
000831- A2 00 40          1530          LDX ##BUFFER POINT TO BUFFER
                          1540 *--------------------------------
000834- E2 20             1550 .1       SEP #$20     CRC=aaaabbbbccccdddd,
                                                           DATA=eeeeffff
000836- A5 01             1560          LDA CRC+1    aaaabbbb .eor. 0000aaaa
000838- 4A                1570          LSR                     = aaaagggg
000839- 4A                1580          LSR
00083A- 4A                1590          LSR
00083B- 4A                1600          LSR          0000aaaa
00083C- 45 01             1610          EOR CRC+1    aaaabbbb
00083E- EB                1620          XBA          AGXX
00083F- A9 00             1630          LDA #0       AG00
000841- C2 20             1640          REP #$20
000843- 4A                1650          LSR
000844- 4A                1660          LSR
000845- 4A                1670          LSR          000a.aaag.ggg0.0000
000846- 85 0A             1680          STA TEMP
000848- 4A                1690          LSR          0000.aaaa.gggg.0000
000849- 45 00             1700          EOR CRC      aaaa.bbbb.cccc.dddd
                                                  = aaaa.gggg.kkkk.dddd
00084B- EB                1710          XBA          kkkk.dddd.aaaa.gggg
00084C- 45 0A             1720          EOR TEMP     000a.aaag.ggg0.0000
00084E- E2 20             1730          SEP #$20
000850- 55 00             1740          EOR 0,X        crchi.crclo
000852- C2 20             1750          REP #$20
000854- 85 00             1760          STA CRC
                          1770 *--------------------------------
000856- E8                1780          INX
000857- E0 02 41          1790          CPX ##LIMIT
00085A- 90 D8             1800          BCC .1
00085C- FB                1810          XCE
00085D- 60                1820          RTS
                          1830 *--------------------------------
                          1840 *   FIND BAD BIT BY BRUCE LOVE'S METHOD
                          1850 *--------------------------------
10-                       1860 DUMMY.CRC   .EQ $10,11
                          1870 *--------------------------------
                          1880 FIND.BAD.BIT
00085E- 20 1B 08          1890          JSR RECV     RECEIVE, COMPUTING NEW CRC
                          1900 *--------------------------------
000861- 18                1910          CLC
000862- FB                1920          XCE          ENTER NATIVE MODE
000863- C2 30             1930          REP #$30     X,M 16 BITS
000865- A2 0F 08          1940          LDX ##$80F   X=BIT NUMBER
000868- A9 01 00          1950          LDA ##1      START DUMMY CRC IN A-REG
00086B- C5 00             1960 .1       CMP CRC
00086D- F0 0B             1970          BEQ .2       ...FOUND BAD BIT!
00086F- CA                1980          DEX          DECREMENT BIT NUMBER
000870- 30 08             1990          BMI .2       ...WENT TOO FAR,
                                                        COULDN'T FIND BAD BIT
000872- 0A                2000          ASL          SHIFT DUMY CRC
000873- 90 F6             2010          BCC .1
000875- 49 21 10          2020          EOR ##$1021
000878- B0 F1             2030          BCS .1       ...ALWAYS
                          2040 *--------------------------------
00087A- 8A                2050 .2       TXA          BIT NUMBER
00087B- 38                2060          SEC
00087C- FB                2070          XCE
00087D- EB                2080          XBA
00087E- 4C 41 F9          2090          JMP PRNTAX
                          2100 *--------------------------------
```

Correction to Fast Garbage Collector...Bob Sander-Cederlof

In the March 1984 AAL, Paul Shetler gave us a very fast garbage collector for Applesoft. Last week Keith Satterley called from Australia, and mentioned he thought there was a bug in the handling of strings over 128 characters long. I looked into it, and he is right.

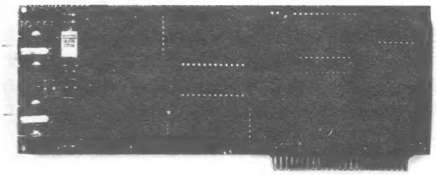
The bug is in the loop in lines 3240-3320, on page 9 of that issue. The loop moves a string from one place in memory to another. The way we printed the code, a string longer than 128 characters would only have one byte moved! Here is the old code and the correct code, side-by-side:

```
-----old code------------        -----correct code--------

3240     LDY STRING.LENGTH       3240     LDY STRING.LENGTH
3250     DEY                     3250 .3  DEY
3260 .3  LDA (FRESPC),Y          3260     LDA (FRESPC),Y
3270     STA (LOWTR),Y           3270     STA (LOWTR),Y
3280     DEY                     3280     TYA
3290     BPL .3                  3290     BNE .3
3300     BMI .1                  3300     BEQ .1
```

Can you see why the new code works and the old doesn't?

Viewmaster 80,™ the sharpest 80 column card of them all.



	BUILT-IN SOFTSWITCH	SHIFT KEY SUPPORT	LOW POWER DESIGN	80 COLUMN HOME	7 x 9 DOT MATRIX	LIGHT PEN INPUTS	40 COLUMN OVERRIDE	INVERSE CHARACTERS	
VIEWMASTER 80	✓	✓	✓	✓	✓	✓	✓	✓	
SUPRTERM		✓					✓	✓	
WIZARD 80					✓		✓	✓	
VISION 80	✓	✓			✓				
OMNIVISION		✓					✓	✓	
VIEWMAX 80	✓	✓			✓			✓	
SMARTERM	✓	✓				✓	✓		
VIDEOTERM		✓	✓		✓	✓		✓	

Now, get great resolution and total software compatibility for your Apple II® or II+®.

One look at the chart will give you some of the reasons there's only one smart choice in 80 column cards for your Apple. But the real secret to Viewmaster 80's success is something even better: Total compatibility.

The Viewmaster 80 works with all 80 column applications, including DOS 3.3, PRO-DOS, CP/M, Pascal, WordStar, Format II, Easywriter, Applewriter II, Supertext 80, Zardax, Apple PI, Letter Perfect, dBASE II, Visicalc, Multiplan, and *hundreds* of others.

And the Viewmaster 80 delivers a super sharp, state-of-the-art display with a 7 × 9 character matrix for clear, easily readable characters. Here are just a few of the powerful features the Viewmaster 80 delivers for a great price ($139):

• 80 Characters by 24 lines • Fully compatible with all Apple languages and software • Super sharp 7 × 9 character matrix with true descenders • Highest compatibility with existing 80 column software • Power and input connector for light pen • Very low power consumption • High speed (18 MHZ) scroll rate • Upper and lower case characters with true descenders, both inverse and normal; all on-screen editing functions are supported • User-definable cursor shape • Compatible with Apple II, II+ and IIe • Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Æ Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

DOS Patch: Prevent Direct Commands.....Richard Gendron

I operate a AE/CATFUR line using my Apple and a modem in Montreal, Quebec. I have found that protecting your DOS from illegal entry can be a tough job to say the least.

In searching for ways to protect my system, I came across an interesting address in DOS: at $A026 there is some code which is executed whenever you try to type in a DOS command. The code checks to see if the command you typed is allowed as a direct command, and if not gives you the NOT DIRECT COMMAND message (or ERROR 15 if you are using DiversiDOS).

I have written a little patch that will catch you when you type a DOS command, and re-RUN the Applesoft program. If a sneaky caller finds a way to get out of the executing Applesoft program, at least he/she will be prevented from doing DOS commands.

Now every lock should have a key. You do want to be able to use your own DOS in direct mode, so I have included a way to turn off the protection. If you type "PRINT USR (0)" the system will respond with "PW:". Then enter a two-character password and the protection patch will be removed. Then you can CATALOG, DELETE, or whatever you want to do.

Since I use Diversi-DOS, and in both the 48K and 64K configurations, I set up my patching program so that it will work with both. The code which checks which version is loaded is in lines 1220-1260 and lines 1390-1410. If the output hook at $36,37 points up to $BDxx or higher, the 64K version must be running. Normal 48K DOS points to $9EBD.

These patches worked on my system, but yours may be a little different depending on which version of DOS you use. Examine carefully all the addresses I use inside DOS to see if yours is the same as mine before you try to use these patches.

```
                  1000 *SAVE GENDRON DOS MODS
                  1010 *-------------------------------
                  1020 *  DOS PROTECTION FOR THE DIRECT COMMAND "ERROR 15"
                  1030 *   WRITTEN BY RICHARD GENDRON FOR USE ON TRANSFERS ][
                  1040 *  (514) 738-1247 (AE/CAT-FUR)
                  1050 *-------------------------------
                  1060          .OR $300
                  1070 *-------------------------------
                  1080 INSTALL
0300- A9 4C       1090          LDA #$4C     BUILD "USR" VECTOR
0302- 85 0A       1100          STA $0A      "JMP" OPCODE
0304- A9 46       1110          LDA #USR
0306- 85 0B       1120          STA $0B
0308- A9 03       1130          LDA /USR
030A- 85 0C       1140          STA $0C
                  1150 *---MOVE DATA INTO DOS-----------
030C- A2 04       1160          LDX #P1-PATCHES       POINT AT OUR PATCHES
                  1170 ***      JMP PATCH.DOS
                  1180 *-------------------------------
                  1190 PATCH.DOS
030E- A9 26       1200          LDA #$A026
0310- 85 00       1210          STA $00
0312- A5 37       1220          LDA $37      48K OR 64K DOS?
0314- C9 BD       1230          CMP #$BD     CARRY CLEAR IF 48K
0316- A9 A0       1240          LDA /$A026   ...48K
0318- 90 02       1250          BCC .1       ...48K
031A- A9 E0       1260          LDA /$E026   ...64K
031C- 85 01       1270 .1       STA $01
```

```
031E- A0 04     1280        LDY #4        MOVE 5 BYTES
0320- BD 71 03  1290 .2     LDA PATCHES,X
0323- 91 00     1300        STA ($00),Y
0325- CA        1310        DEX
0326- 88        1320        DEY
0327- 10 F7     1330        BPL .2
0329- 60        1340        RTS
                1350 *--------------------------------
                1360 REBOOT
032A- 78        1370        SEI             TURN OFF INTERRUPTS
032B- 20 EA 03  1380        JSR $03EA       RESET THE I/O HOOKS
032E- A5 37     1390        LDA $37         WHICH DOS?
0330- C9 BD     1400        CMP #$BD        IS IT 64K DOS ?
0332- 30 0C     1410        BMI .1          NO IT IS NOT
0334- 2C 81 C0  1420        BIT $C081       YES IT IS, SO TURN
0337- 2C 81 C0  1430        BIT $C081        OFF LANGUAGE CARD
033A- 20 16 E3  1440        JSR $E316       DOS "CLOSE" ALL FILES
033D- 4C 66 D5  1450        JMP $D566       APPLESOFT "RUN"
0340- 20 16 A3  1460 .1     JSR $A316       DOS "CLOSE" ALL FILES
0343- 4C FC A4  1470        JMP $A4FC       48K "DOS RUN"
                1480 *--------------------------------
                1490 USR
0346- A0 00     1500        LDY #0
0348- B9 67 03  1510 .1     LDA PASSWORD,Y GET PASSWORD TEXT
034B- F0 06     1520        BEQ .2          ...END OF STRING
034D- 20 6B 03  1530        JSR PRINT       NO, SO PRINT IT
0350- C8        1540        INY
0351- D0 F5     1550        BNE .1          ...ALWAYS
                1560 *--------------------------------
0353- 20 6E 03  1570 .2     JSR INPUT       GET A KEY
0356- C9 AA     1580        CMP #"*         "*" ?
0358- D0 0C     1590        BNE .4          NO, SO BYE BYE
035A- 20 6E 03  1600        JSR INPUT       YES, SO GET ANOTHER
035D- C9 AE     1610        CMP #".         "." ?
035F- D0 05     1620        BNE .4          NO, SO BYE BYE
                1630 *--------------------------------
0361- A2 09     1640        LDX #P2-PATCHES
0363- 20 0E 03  1650        JSR PATCH.DOS
0366- 60        1660 .4     RTS             WE HAVE FINISHED
                1670 *--------------------------------
                1680 *   TEXT TO BE PRINTED WHEN A
                1690 * "PRINT  USR(0)" COMMAND IS DONE
                1700 *        IN APPLESOFT
                1710 *--------------------------------
                1720 PASSWORD
0367- D0 D7 BA  1730        .AS -"PW:"
036A- 00        1740        .HS 00
                1750 *--------------------------------
                1760 *  INPUT AND PRINT SUBROUTINES
                1770 *--------------------------------
036B- 6C 36 00  1780 PRINT  JMP ($36)
036E- 6C 38 00  1790 INPUT  JMP ($38)
                1800 *--------------------------------
                1810 PATCHES
0371- 4C 2A 03  1820        JMP REBOOT    CALL OUR NEW CODE
0374- EA        1830        NOP           NEEDED FOR DIVERSI-DOS
0375- EA        1840 P1     NOP
                1850 *--------------------------------
0376- A9 02 39
0379- 09        1860        .HS A9023909    ORIGINAL CODE
037A- A9        1870 P2     .HS A9
                1880 *--------------------------------
```

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $18 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada, and Mexico; add $14 postage for other countries. Back issues are available for $1.80 each (other countries add $1 per back issue for postage).